Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT®
Discrete Applied Mathematics 137 (2004) 159–171
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Spanners and message distribution in networks☆

Arthur M. Farley[a], Andrzej Proskurowski[a], Daniel Zappala[a], Kurt Windisch[a,b]

[a] *Computer and Information Science Department, University of Oregon, Eugene, OR 97403, USA*
[b] *Advanced Network Technology Center, Computing Center, University of Oregon, Eugene, OR 97403, USA*

Received 30 October 1999; received in revised form 6 July 2001; accepted 19 October 2002

Abstract

We investigate the applicability of spanners as substructures that offer both low cost and low delay for broadcast and multicast. A k-spanner has the potential to offer lower delay than shared multicast trees because it limits the distance between any two nodes in the network to a multiplicative factor k of the shortest-path distance. Using simulation over random topologies, we compare k-spanners to single-source minimum-distance spanning trees and show that a k-spanner can have similar cost and lower delay. We illustrate that by varying the value of k a spanner can be made to gradually favor cost over delay. These results indicate the spanner can reduce the cost of flooding, which is a basic mechanism used in multicast routing protocols.

Keywords: Networks; Broadcast; Flooding; Graph; k-spanner; Simulation

1. Introduction

Several current multicast routing protocols use flooding as a means of initially reaching the members of a multicast group. This flooding represents one of the major costs of multicast routing.

We are studying substructures that can limit this cost while bounding the resulting increase in delay. In particular, we are studying k-spanners, subgraphs that guarantee that the distance between any two nodes does not exceed the shortest path distance

☆ Research of all authors supported in part by National Science Foundation grants NCR-97-14680 and ANI-99-77524.
E-mail address: andrzej@cs.uoregon.edu (A. Proskurowski).

0166-218X/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0166-218X(03)00259-2

by a multiplicative factor k. Unlike other commonly used structures, such as Steiner minimal tree approximations [17,14,10] or core-based trees [1,7], the k-spanner can bound the path length between any two vertices in the graph. This bound can take into account hop count, Euclidean distance, or link weights.

In this paper, we are concerned primarily with broadcast, rather than multicast, because algorithms for spanners (where all nodes are group members) are more tractable than algorithms for substructures where only a subset of nodes are members. Broadcast is also the basis for bootstrapping multicast communication in several multicast protocols, namely DVMRP [13] and PIM [7]. We have developed similar protocols that bootstrap multicast by flooding a spanner. More details on our planned future work with multicast are given in Section 6.

Our primary focus in this paper is a performance evaluation of spanners relative to shortest-path spanning trees. While the theoretical bounds of k-spanners are well known, their average behavior is not. Our goal is to characterize the average behavior of k-spanners as a function of k to assess their applicability for group communication. We limit our investigations to light-weight spanners, which approximate minimum-weight spanning tree within $O(\log n)$ [11]. We demonstrate through simulation that a k-spanner can be made to favor low cost (the number of edges) or low delay (average and maximum distances) by adjusting k.

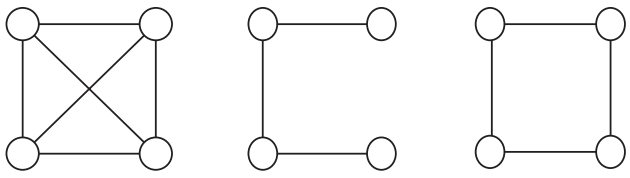
The rest of the paper is organized as follows. Section 2 defines graph spanners and reviews relevant theoretical results. We then discuss in Section 3 how graph spanners can be used for group communication. In Section 4, we present our experiment design for studying spanner characteristics and in Section 5 we present our results. Finally, in Section 6 we discuss our conclusions and several areas for future research, particularly in applying this work to multicast.

2. Graph spanners

The notion of graph spanners was introduced about a decade ago in [12]. Each spanner is parameterized by a constant k, as follows: A k-spanner of a graph $G=(V,E)$ is a graph $S=(V,E')$, where E' is a subset of E, such that the distance in S between any pair of vertices x and y of V is not more than k times the distance between x and y in G. The *distance* between two vertices of a graph is defined as the minimum, over all paths connecting the two vertices in the graph, of the sum of edge lengths on the path.

To illustrate the differences between spanning trees and spanners, we show an example of a minimum distance spanning tree and a 2-spanner of a 4-vertex graph in Fig. 1.

A k-spanner is formed by removing certain edges from G while guaranteeing that the distance between any pair of vertices is not stretched by more than the multiplicative factor k. Unfortunately, the problem of deciding whether a k-spanner exists that has total weight (i.e., sum of weights associated with edges) less than W in an arbitrary graph G has been shown to be *NP*-complete for most values of k [3]. The problem remains *NP*-complete for any planar, biconnected graph G, with lower limits for k depending on whether edges of G are weighted or unweighted (i.e., weights all equal



(a) Graph (b) Minimum Spanning Tree (c) 2-Spanner

Fig. 1. Examples of a spanning tree and a 2-spanner of a 4-vertex graph.

to one). Finally, determining whether an arbitrary graph has a k-spanner that is a tree or that is planar is also *NP*-complete [4,2].

Given the apparent intractability of finding minimum-weight spanners, a greedy algorithm that finds relatively light-weight spanners can be used. The algorithm is defined as follows:

Algorithm LightWeightSpanner(G, k)
// Input: an arbitrary connected graph $G = (V, E)$ and an integer $k > 1$
// Output: a k-spanner $S = (V, E')$ of G

Step 0. Let E' be empty.
Step 1. Sort the edges E in increasing order of weight.
Step 2. For each edge $e = (x, y)$ in E (considered in sorted order)
    if { the distance between x and y in $S = (V, E')$
        is greater than k times the distance in G }
    then { add edge e to E' }
Step 3. Report $S = (V, E')$ as result.

The algorithm considers light edges first and only adds an edge to the spanner when it is necessary to connect neighbors so as to maintain the given stretch factor. The resultant graph is a k-spanner because, if each edge in G has not been stretched by more than k, then any path between nonneighbors in G has not been stretched by more than k. The algorithm is clearly implementable in polynomial time. Mansour and Peleg have determined some properties of the spanners generated by this algorithm. For a graph with n vertices, the algorithm returns a sparse (with $O(n)$ edges) and light-weight (within $O(\log n)$ of the minimum weight of a spanning tree) k-spanner, with $k = O(\log n)$ [11].

Throughout this paper we use light-weight spanners generated by the above algorithm. All our results refer to these spanners.

3. Broadcast and multicast over spanners

How might spanners be employed for message distribution in networks? One potential application is as a virtual topology for broadcast and multicast. *Broadcast* is the

communication process whereby a message is sent from one sender to all other sites of a network, such as an administrative domain of the Internet. A need for broadcasting can arise in a number of network management and control contexts, as well as in implementations of distributed algorithms. *Multicast* involves sending a message to a subset of the sites in a network, and is typically used for group communication, such as audio or video conferencing.

A common method used to perform broadcast in a network is *reverse path flooding*. The flooding process begins when a source transmits a message by sending it on all of its links. Each router that receives the message checks whether it was received on the link that it would use to *send* messages to the source along its shortest-path route. If this check succeeds, then the router forwards the message on all other links, otherwise it drops the message. Hence, with this process a message is flooded along the *reverse* of the shortest paths used to reach a source. This method of flooding is used by both DVMRP [13] and PIM Dense Mode [6].

In order to perform reverse-path forwarding in a spanner, each router must be able to know which of its links may be used to reach a given source via its shortest path. In the case of DVMRP and PIM Dense Mode, a unicast routing protocol provides this information. When using a spanner, however, this routing information is no longer valid, since many of the links in the original graph have been "deleted." Hence, some new mechanism must be used to provide shortest paths in relation to the spanner.

In this vein, we have proposed two methods for performing reverse-path flooding in a spanner, based on the two common methods for routing in packet-switched networks. The first of these methods assumes that a link-state protocol is used for routing packets in the network. In this case, routers run a centralized version of the light-weight spanner algorithm and can likewise run Dijkstra's shortest-path algorithm to determine paths within the spanner. The second method is intended for networks with distance-vector routing and uses a distributed algorithm to compute the spanner. Routers then use "hop count" to determine the shortest path to a source. Details of these protocols can be found in [8].

The algorithm for sending multicast messages in a spanner uses broadcast in a two-step bootstrap mechanism. This mechanism is no different from those used by both DVMRP and PIM. The first step in sending a multicast message is to flood messages along the reverse-path tree to all nodes. In the second step, leaves that are not members of the multicast group may remove themselves from the tree in a process known as pruning. As the pruning occurs, routers in the tree that are left without any children can likewise prune themselves. Eventually, the tree delivers messages only to those nodes that wish to receive them.

Since both multicast and broadcast use reverse-path flooding, the cost of the flooding algorithm is a primary concern. Note that during the flooding, a message traverses every edge at least once and some edges twice. Thus, the number of edges in a graph is a good approximation to the amount of traffic generated by a broadcast through flooding. Using a light-weight k-spanner, we can reduce the cost of flooding by decreasing the number of edges used for broadcast. In the following section, we illustrate the tradeoffs between cost and delay that result from using k-spanners.

4. Experiment design

In order to be useful for bootstrapping multicast, a k-spanner must first have desirable broadcast properties. To assess the utility of broadcasting over spanners, we have designed a set of experiments to measure average cost and delay over a set of random topologies. We compare flooding over light-weight spanners to flooding over the entire network and to flooding over a shortest-path spanning tree rooted at a random node. Flooding over a network gives maximum cost and minimum delay, whereas flooding over a shortest-path spanning tree reduces cost but increases the delay. We have considered this latter style of communication because it has been suggested as the basis for forming a low-cost core-based tree [7]. Details on how other types of shared trees perform based on these two metrics may be found in [16].

We want the randomized topologies to be representative of topologies for administrative domains of the Internet. As such, we generate graphs having either 64 or 128 vertices and average vertex degrees of approximately 4 or 8. In addition, we use two vertex connection schemes: one representing a strictly random adjacency pattern and a second preferring "nearby" vertices over "distant" connections. This results in eight different classes of random graphs, one for each combination of the above three conditions. Our test sets consist of 50 graphs for each type. We generated three sets of graphs for each type to get some indication as to the stability of our results relative to the random number generator used to create the network graphs.

We use the suite of random graph generators available at Georgia Tech [5]. These algorithms all place a set of n vertices at random, uniformly distributed over a square (with diameter L, see below) in the plane, and then consider each pair of vertices in turn, deciding whether an edge is to be added between them. The purely random scheme uses an equal probability of an edge for all pairs of vertices; the locality preference scheme has the probability decreasing exponentially with the distance between the two vertices on the plane [18]. While there has been a discussion of how to model the structure of the Internet at large, the Waxman model is widely regarded as a reasonable model of a single administrative domain (see [18]). The locality method we used is the basic Waxman model, [15] in which the probability of an edge being added between two vertices is $\alpha e^{-d/(\beta L)}$ where d is the distance between the vertices on the plane and α and β are parameters of the method. An increase in α increases the number of edges, and an increase in β increases the ratio of "long" to "short" edges. We use the following parameter settings: for degree 4, 64 vertex graphs, $\alpha = 0.42$, $\beta = 0.14$; for degree 4, 128 vertex graphs, $\alpha = 0.21$, $\beta = 0.14$; for degree 8, 64 vertex graphs, $\alpha = 0.85$, $\beta = 0.15$; and for degree 8, 128 vertex graphs, $\alpha = 0.42$, $\beta = 0.14$. We chose the low value of β to favor short edges (hence "locality") and adjusted α to obtain the desired average degree. (Setting of β follows that of [18].)

The primary metrics we are interested in are number of edges, average distance, and maximum distance in a given graph, its spanners, and spanning tree. We consider three measures for the distance along an edge in a given graph. The first is uniform over all edges; we assume its value is one. Given this measure, the distance between two vertices equals the number of edges ("hop count") in a shortest path between the pair. The second measure is the (Euclidean) distance between the end-vertices of the

edge wrt. their location on the plane (determined when the graph is generated). The third measure is a random weight, assigned from the set $\{1,2,4,8,16\}$, reflecting the fact that for traffic management purposes in networks like the Internet, the cost of an edge may be set to an arbitrary value irrespective of its physical length. The set of weights has been chosen to represent different orders of magnitude. We refer to these three distance measures as the hop, length, and weight measure, respectively.

For each given graph G, we determine the following subgraphs: k-spanners $S_k(G)$, $2 \leqslant k \leqslant 7$, and a minimum-distance spanning tree $T(G)$ from vertex labeled 0, for all three edge-distance measures (a total of 21 subgraphs). Note that each subgraph has the same vertex set as the original graph, but has only a subset of the edges. We then compare each subgraph to the original graph in terms of three metrics: number of edges E, diameter D, and average distance A between vertices. We make these comparisons in terms of the edge-distance measure (either hop, length, or weight) used in constructing the spanner.

We compare subgraphs by computing the ratios of parameter values for a subgraph to the values for the given graph; for example, $E(S_2(G))/E(G)$ would be the ratio of number of edges in a 2-spanner of G to the number of edges in a given graph G. We use the notation $G_{v,d}$ to represent the set of graphs with v vertices and average degree d; for example $G_{64,4}$ represents the set of graphs with 64 vertices and average degree 4.

We generated three test sets of 50 graphs for each of the above profiles. Our results indicate little or no difference between the three test sets for the same parameter settings. Thus, we report the results from the first set of graphs for each parameter combination.

5. Experiment results

The experiment as designed above, generates a complex array of results. There are eight types of random graphs, based on number of vertices, average vertex degree, and whether there was a locality preference in generating edges. Then there are 3 edge-distance measures applied to edges: hop, length, and weight. For each graph and distance measure, we compute k-spanners for k from 2 to 7 and minimum-distance spanning trees rooted at an arbitrary vertex. The metrics are computed for all graphs are: number of edges, average distance between vertices, and greatest distance between vertices (i.e., graph diameter).

The results of our experiment are expressed in terms of the ratios that compare values of the metrics for a spanner or a spanning tree to the corresponding values for the original graph. This use of ratios to compare graph metrics is similar to that used in [16] to evaluate options for multicast shared trees. The ratios of the numbers of edges are less than 1.0, while distance-related ratios are greater than 1.0 (reflecting the increased distance due to excluding some edges). We plot the values of these metric ratios for k-spanners as functions of k ranging from 2 to 7. Each plot presents the average ratio, with error bars indicating the 25th and 75th percentile value, over the 50 graph sample. The ratio of the corresponding parameter value for spanning trees is shown on each plot for comparison purposes.

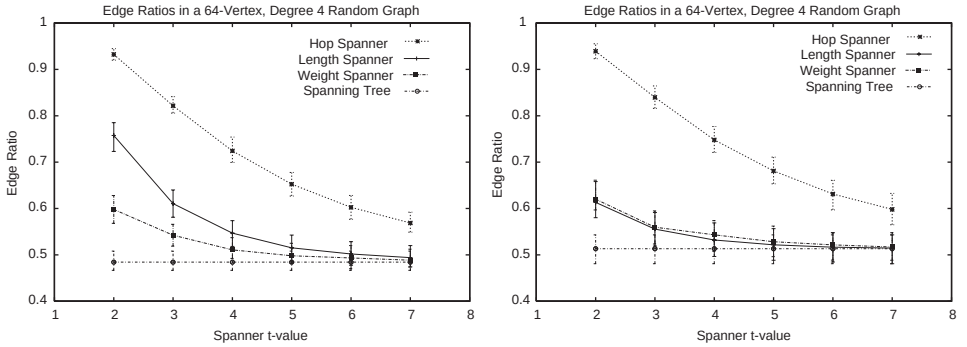


Fig. 2. Edge ratios for the two random graph models (purely random edge placement on the left, locality preference on the right).

5.1. Number of edges

Given a graph with n vertices every spanning tree has $n-1$ edges. As such, the edge ratio for this parameter is a simple function of the average degree of the original graph (i.e., about twice its inverse). For a random graph with average degree 4, we expect the edge ratio to be $(n-1)/2n$, or a little less than 0.50; for degree 8 graphs, we expect $(n-1)/4n$, or a little less than 0.25. There will be some variability due to the randomness of the graphs generated, more so for the locality preference graphs as the parameter settings that influence average degree were determined by trial and error. The spanning trees set the lower bound for the number of edges in the spanners. Edge ratios for the spanners will range from a possible high of 1.0 (all edges included) down to the ratios associated with the spanning trees.

Let us start with results for $G_{64,4}$, the 64 vertex graphs having vertices with average degree 4. Fig. 2 shows two diagrams depicting edge ratios for k-spanners with k from 2 to 7; one graph showing the values of metrics for purely random graphs and the other showing the same values for graphs with a locality preference. We first note the general shape of the curves. The number of edges in k-spanners approach the number of edges in the spanning tree as k increases. This is consistent with the theoretical results mentioned earlier regarding properties of the light-weight spanners generated by the spanner algorithm we used. The decrease in number of edges is especially pronounced for the length and weight edge-distance measures. In addition, the numbers of edges in k-spanners of locality preference graphs for these two distance measures have nearly identical behavior as k varies.

The k-spanners constructed for the hop edge-distance measure (*hop spanners*) consistently have more edges than those constructed for length and weight edge-distance measures (*length* and *weight spanners*). This phenomenon has a straightforward explanation. Recall the algorithm only adds edges (considered in increasing order of their length or weight) to the spanner as needed to maintain an increase in distance by a factor of less than k between neighbors in the original graph. A "long" edge in the

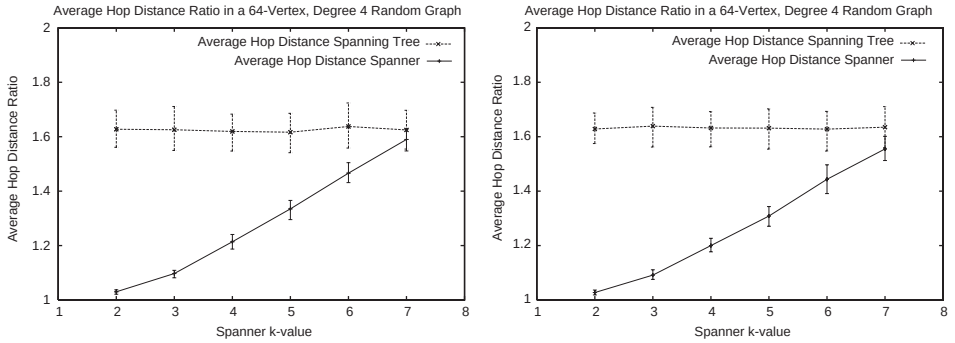


Fig. 3. Average hop distance for the two random graph models (purely random edge placement on the left, locality preference on the right).

length or weight spanners may not be needed, as there may be already in the spanner a path of (possibly more than k) 'shorter edges between the end-vertices of that edge. In a hop spanner, all edges have equal distance and are considered in a random order. Thus, an edge can only be omitted from the hop spanner when a path of k or fewer edges between its end-vertices has already been considered, which can be expected to happen less often.

As far as any differences between purely random and locality preference graphs, edge ratios for the hop and weight spanners are affected little, if at all. For the length edge-distance measure, some differences do appear. The edge ratio for length spanners is consistently lower in the locality preference graphs. This indicates that the locality preference for connecting neighbors in the given, randomly generated graph does provide a small advantage when forming spanners based on the length distance measure in terms of number of edges required. As there are more "short" edges in such graphs, a higher percentage of what "long" edges there are can be eliminated by the spanner algorithm. Note that when all edges are considered to have the same edge distance, as with the hop measure, or when edge distances do not correspond to the basis for the locality preference, as with the weight measure, there is essentially no effect on number of edges.

5.2. Distance measures

We next turn to distance-related results for the graphs with 64 vertices and average degree 4. As spanners have fewer edges than the original graphs, distances between vertices will increase. By how much they increase and how they compare to increases for minimum-distance spanning trees are the key questions. While the number of edges is related to traffic generated during message distribution, distance corresponds roughly to communication delay in the network. Let us first consider the hop distance measure, as this is usually considered to be most relevant to distance and delay in the Internet.

First, we consider average hop distance between vertices. Fig. 3 presents results for the hop spanners, the first plot for purely random graphs and the second one for

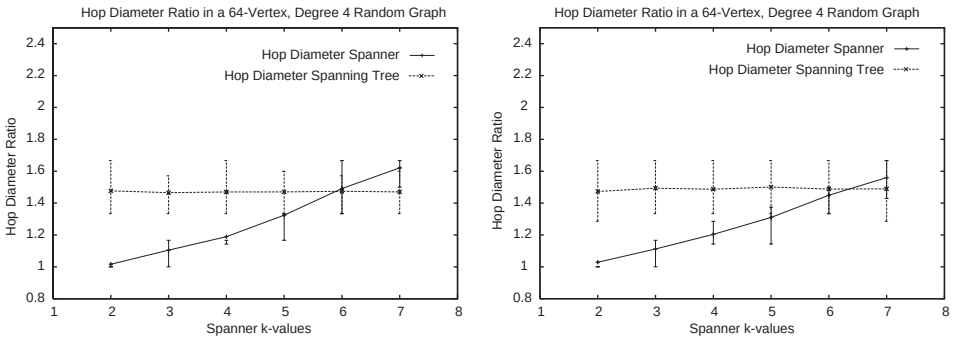


Fig. 4. Average hop diameter for the two random graph models (purely random edge placement on the left, locality preference on the right).

locality preference graphs. In a minimum-distance spanning tree the average distance ratio between the tree and the original graph, $A(T(G_{64,4}))/A(G_{64,4})$, is slightly greater than 1.6. We see that average hop distance increases by over 60%, regardless of locality preference in neighbor selection. Recall that this is for group communication using the spanning tree as a shared structure, so we are considering all nodes, not just the root of the tree, as potential sources.

For a 4-spanner of such graphs, the ratio $A(S_4(G_{64,4})/A(G_{64,4}))$ is approximately 1.2, representing only a 20% increase in average hop distance. As spanners do not approximate trees for small k, there are alternative paths that provide shortcuts between vertices, resulting in average distances that are significantly lower in such spanners. We see average hop distances are not impacted by locality preference in the network topology.

In Fig. 4, we consider hop diameters in degree 4 graphs, again by two plots, one for purely random graphs and one for locality preference graphs. In a minimum-distance spanning tree, the ratio $D(T(G_{64,4}))/D(G_{64,4})$ is about 1.5, representing a 50% increase over hop diameters of the original graph. For a 4-spanner of such graphs, the ratio $D(S_4(G_{64,4})/D(G_{64,4}))$ is approximately 1.2, for only a 20% increase in hop diameter. Again, we see diameters associated with hop distances are not changed by locality preference in modeling the network topology. Overall, we see 4-spanners perform very well in comparison to minimum-distance spanning trees when considering average hop distance; the difference is less for the hop diameter measure, but still significant.

Our results show that, as was true for number of edges, only results associated with the length measure for edge distance are impacted by locality preference in forming edge connections. Fig. 5 shows how average distance ratios in spanners and spanning trees for the length edge-distance measure is indeed improved in the locality preference networks. Even though the number of edges in spanners of the locality preference graphs is less than in spanners for the pure random model, their diameters and average distances are also smaller. Thus, if locality preference based on the edge-distance measure plays a role in creating a network graph, then spanners of that graph perform even better than indicated by the results for hop spanners.

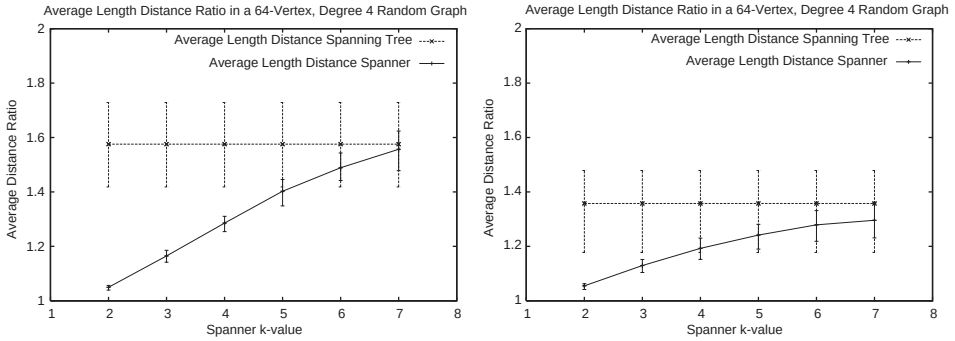


Fig. 5. Average length distance ratios (purely random edge placement on the left, locality preference on the right).

Note that the distance metrics for spanning trees do not represent an upper limit on those metrics for k-spanners. Spanners do not focus on minimizing any distances, only limiting maximum distance growth. Spanners for higher values of k tend to permit greater distances between vertices than do the minimum-distance spanning trees. This is especially noticeable when considering the diameter measure. The k-spanners we generated have higher hop diameter than the corresponding spanning trees when k was greater or equal to 5.

5.3. Vertices and degrees

As average vertex degree increases, both spanning trees and spanners have lower edge ratios. In Fig. 6, we turn our attention to $G_{64,8}$, graphs having 64 vertices with average degree of 8. In Fig. 2 we saw that for a degree 4 graph, a 4-spanner has less than 55% of the edges of the original graph for the length and weight edge-distance measures; a 4-spanner for the hop edge-distance measure has about 75% of the edges. Now, in a degree 8 graph, a 4-spanner has only approximately 30% of the edges of the original graph for the length and weight edge-distance measures; for the hop edge-distance measure, it has less than 50%. Recall that if we use a spanner to broadcast a message by flooding within a domain, the number of edges corresponds roughly to the message traffic generated. We can see that using a 4-spanner to flood the network results in significant reduction in traffic, and that this benefit increases as average degree of vertices in the network graph increases.

When considering the effects of average vertex degree on distance measures, we find that the average hop distance ratio for the spanning tree is approximately 1.75 in graphs with average degree 8; for 3-spanners, it is less than 1.25, and for 4-spanners it is less than 1.45, regardless of locality preference. In these same graphs, the hop diameter ratio for the spanning tree is approximately 1.55; for 3-spanners it is less than 1.25, and for 4-spanners it is less than 1.50, regardless of locality preference. We see that distance ratios tend to increase overall, reflecting a penalty incurred by being able to discard more edges when forming the spanners in higher degree graphs. The

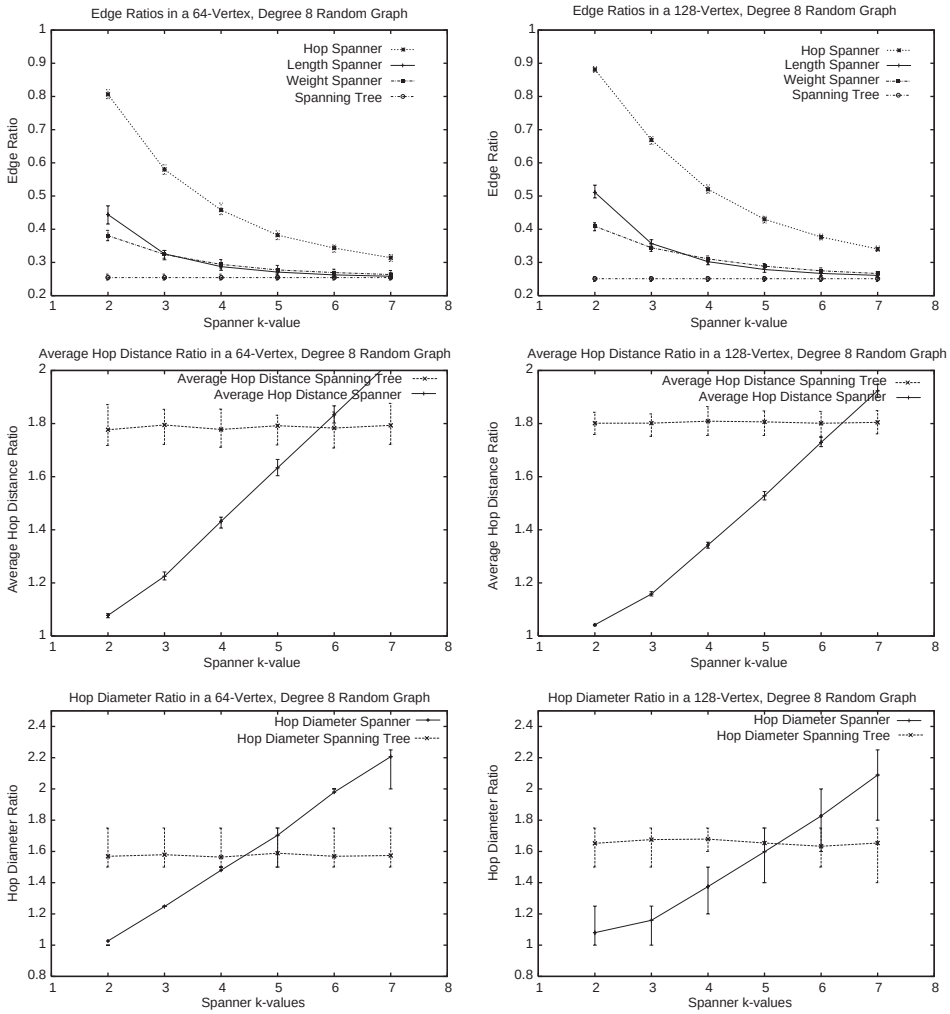


Fig. 6. Representative results for degree-8 graphs.

pattern of results we find when comparing spanners to the minimum-distance spanning trees are essentially unchanged, however. The 3- or 4-spanners significantly reduce distances over the spanning tree. As in degree-4 graphs, spanners show better relative performance on the average distance metric than on the diameter metric. Fig. 6 shows some representative results for spanners of degree-8 graphs.

Finally, we consider network graphs having 128 vertices. The first thing we note is that the general pattern of results remains unchanged. Overall, the edge ratios tend to be slightly higher for the spanners. This increase in edge ratios produces slightly better distance-related ratios in the spanners. The results for graphs with 128 vertices

are somewhat more supportive of the usefulness of 3-, 4-, or 5-spanners as distribution topologies; spanner distance-related ratios are slightly better when compared to minimum-distance spanning tree results.

6. Conclusions

In this paper, we conduct what we believe to be the first experimental analysis of k-spanners with respect to cost and delay. We have shown the k-spanner to be an intermediate between flooding over the entire network and using a single-source, minimum-distance spanning tree. Using a value of 4 or 5 for k produces a k-spanner with cost approaching that of the spanning tree, yet with significantly lower delay. These characteristics enable the spanner to reduce the cost of reverse-path flooding, which is a basic mechanism used in multicast routing protocols. As a result of this work we have developed several protocols for distributing multicast messages over a spanner [8].

As part of continuing research, we have defined a distributed protocol for computing k-spanners. With only local knowledge of network topology, sites send messages to nearby sites and determine *alternate paths* for an edge, being paths from one end of the edge to the other of length less than or equal to k times the length of the edge. Sites then propose deletion of incident edges having alternate paths; these are accepted by nearby sites if removal of the edge would not eliminate all alternate paths associated with an edge already deleted. Experiments indicate the resultant spanners exhibit cost and delay properties similar to those of the light-weight spanners considered in this report [9]. This protocol is an important step in making spanners more applicable to network communication, as the spanners need not be computed centrally.

We have also been considering multicast protocols over shared structures that have multiple cores or rendezvous points. Sites join a group by selecting a nearest core associated with the group and joining to a distribution tree rooted there. What is needed in such a scheme is a shared structure for communication amongst the core sites. We have proposed and implemented a prototype protocol that uses a k-spanner amongst the core sites for this purpose. We call such a structure a *k-stanner*, being related to a Steiner tree that only spans a subset of sites. We have defined a distributed, 1-stanner protocol, which when run on top of a previously determined k-spanner, determines a k-stanner amongst the cores. We have called the combined spanner/stanner multicast protocol *DomCast*. We select sites in the core to form a type of *dominating set* of all sites in a group. To send a message, a site sends a message to a nearest core, which then distributes the message to the cores over the stanner; each core then distributes the message over the multicast tree that it roots for the group [9].

In conclusion, spanners offer an interesting alternative to trees as shared substructures for particular communication tasks, such as multicast. By varying k, such structures can be tuned to reflect the relative importance of cost and delay for a given application. The work reported here reflects our initial experimentation with spanners; we are continuing research exploring their applicability to communication tasks of interest.

Acknowledgements

We would like to thank Iyer Sivaramakrishna for producing appealing plots from an unappealing slew of numbers, and Nita Viswanath for running the program producing those numbers.

References

[1] A.J. Ballardie, P.F. Francis, J. Crowcroft, Core based trees, in: Proceedings of ACM SIGCOMM, London, August 1993.
[2] U. Brandes, D. Handke, NP-Completeness results for minimum plannar spanners, Discr. Math. Theoret. Comput. Sci. 3 (1998) 110.
[3] L. Cai, NP-completeness of minimum spanner problem, Discrete Appl. Math. 48 (1994) 187–194.
[4] L. Cai, D.G. Corneil, Tree spanners, SIAM J. Discrete Math. 8 (1995) 359–387.
[5] K. Calvert, E. Zegura, Georgia Tech Internetwork Topology Models, Software at http://www.cc.gatech.edu.
[6] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, L. Wei, Protocol Independent Multicast Version 2 Dense Mode Specification, Internet Draft: work in progress, June 1999.
[7] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, L. Wei, An architecture for wide-area multicast routing, in: Proceedings of ACM SIGCOMM, San Francisco, August 1994.
[8] A. Farley, A. Proskurowski, K. Windish, Spanners and message distribution in networks, Technical Report UO-TR-99-02, University of Oregon, 1999.
[9] A. Farley, A. Proskurowski, D. Zappala, Domcast: a multicore multicast protocol, Technical Report UO-TR-01-02, University of Oregon, 2001.
[10] L. Kou, G. Markowsky, L. Berman, A fast algorithm for Steiner trees, Acta Inform. 15 (1981) 141–145.
[11] Y. Mansour, D. Peleg, An approximation algorithm for minimum-cost network design, Technical Report CS94-22, Weizmann Institute of Science, Faculty of Mathematical Sciences, January 1, 1994.
[12] D. Peleg, J.D. Ullman, An optimal synchronizer for the hypercube, in: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, 10–12 August 1987, pp. 77–85.
[13] D. Waitzman, C. Partridge, S. Deering, Distance vector multicast routing protocol, RFC 1075, November 1988.
[14] D.W. Wall, Mechanisms for broadcast and selective broadcast, Ph.D. Thesis, Department of Electrical Engineering, Stanford University, 1980.
[15] B.M. Waxman, Routing of multipoint connections, IEEE J. Selected Areas Comm. 6 (9) (1988) 1617–1622.
[16] L. Wei, D. Estrin, The trade-offs of multicast trees and algorithms, in: Proceedings of 1994 International Conference on Computer Communications Networks, San Francisco, September 1994.
[17] P. Winter, Steiner problem in networks: a survey, IEEE Networks 17 (2) (1987) 129–167.
[18] E.W. Zegura, K. Calvert, S. Bhattacharjee, How to model an internetwork, in: IEEE INFOCOM, San Francisco, 1996.